

# Software-Defined Underwater Acoustic Networking Platform

Dustin Torres, Jonathan Friedman, Thomas Schmid, Mani B. Srivastava  
Networked and Embedded Systems Laboratory  
Electrical Engineering Department  
University of California, Los Angeles

## ABSTRACT

Currently acoustics is the primary modality for underwater communication even though it presents a difficult channel. To try and cope with the challenges of the channel many MAC protocols and PHY layer techniques have been proposed. In this paper we present a research platform that allows developers to easily implement and compare their protocols in an underwater network and configure them at runtime. We have built our platform using widely supported software that has been successfully used in terrestrial radio and network development. The flexibility of development tools such as software defined radio have provided the ability for rapid growth in the community. Our platform adapts some of these tools to work well with the underwater environment while maintaining flexibility, ultimately providing an end-to-end networking approach for underwater acoustic development.

## 1. INTRODUCTION

The underwater medium presents many challenges for digital communication. There is limited available bandwidth and high bit error rates caused by multipath, fading, and long propagation delay. The speed of sound is five orders of magnitude less than that of terrestrial radio, which can make it hard for underwater networks to synchronize, exchange data, update routes, and communicate efficiently.

Due to the severity of multipath underwater, a receiving node might be at a point where there is little energy in the received signal making it hard to receive without errors [14]. These spots of destructive multipath interference vary with time due to the movement of water, and make it quite hard to even have a static network topology. Along with multipath, other unfavorable characteristics such as ambient noise, bubbles, surface scattering, and slow propagation speed make developing underwater networks a difficult task [14].

Furthermore, the ocean varies significantly both temporally

and spatially, which makes it challenging to create a model of a “typical” underwater acoustic channel [14]. Since there is no “typical” model, there is no single network architecture that works best in all situations. Therefore, depending on the current characteristics of the channel, there is a variable amount of bandwidth, various noise sources in different frequency bands, varying inter-symbol interference (ISI) depending on the water depth, and many other characteristics under consideration. Even if the channel noise is known, attenuation still depends on both distance and frequency, along with space and time varying multipath [16].

We present a software defined Underwater Acoustic Networking platform (UANT) to aid development of underwater acoustic networks. A software defined system can help to address the constantly changing underwater acoustic channel by way of reconfigurability. A flexible approach allows for system parameters at all layers to be easily modified without the need for specialized hardware. UANT uses GNU Radio, a software defined radio framework, to achieve configurability at the physical layer. TinyOS has been widely adopted for the use on sensor network platforms and provides a full network stack. We adapted these widely supported tools that have proven effective prototyping, development, and implementation for terrestrial networks to be used in UANT.

Since the characteristics of the underwater acoustic channel cannot be properly modeled with a static configuration, it is important to be able to change the properties of an acoustic modem at run time. UANT has the flexibility of software defined radios and the advantages of the network layers of TinyOS and Linux, ultimately providing an end-to-end network for easy underwater development from the physical to application layer.

## 2. BACKGROUND

Software defined techniques have been of interest in recent years not only for terrestrial radios but also underwater acoustics. In [6], Jones describes some of the benefits to the use of software defined radio for underwater use. She talks about the possibility to improve underwater acoustic performance by using methods that have worked well with terrestrial radios such as Cognitive radio via software defined techniques.

The rModem [17] developed at MIT by Sozer and Stojanovic, was created with a similar goal of having a configurable

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WUWNet'09, November 3, 2009, Berkeley, CA, USA.  
Copyright 2009 ACM 978-1-60558-821-6 ...\$10.00

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE <b>NOV 2009</b>		2. REPORT TYPE		3. DATES COVERED <b>00-00-2009 to 00-00-2009</b>	
4. TITLE AND SUBTITLE <b>Software-Defined Underwater Acoustic Networking Platform</b>				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>University of California,Electrical Engineering Department,Networked and Embedded Systems Laboratory,Los Angeles,CA,90092</b>				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release; distribution unlimited</b>					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT <b>see report</b>					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT <b>Same as Report (SAR)</b>	18. NUMBER OF PAGES <b>8</b>	19a. NAME OF RESPONSIBLE PERSON
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>			

acoustic modem. However, the rModem is a standalone board that uses an FPGA and DSP. Furthermore, rModem focuses on lower layers and does not provide an end-to-end networking environment, making it hard to study the impact on actual applications. We hope to supplement platforms such as the rModem by aiding research in MAC and PHY layer protocols that can ultimately be implemented on standalone nodes to create an underwater network.

## 2.1 GNU Radio and USRP

Software defined radio has received a lot of attention most notably in the research community. The ability to use software to modulate and manipulate the received and transmitted signals allows for rapid development without the need or cost of specialized hardware. GNU Radio [4], one of the most popular SDR frameworks, is comprised of a flow graph and signal processing blocks. The signal processing blocks are written in C++ and act as the “heavy lifters” whereas the flow graph is setup in Python in order to move data from one block to the next. In this way many modulation schemes can be created using standard C++ blocks (already included in GNU Radio) and connecting them together in a flow graph. There is a large community of users who have contributed to this open source project, both signal processing blocks as well as various applications. The many contributions have led to a large library of modulation schemes including GMSK, PSK, QAM, CPM, OFDM, and more.

We have modified one of the digital communication applications that is included with the GNU Radio distribution. The application was created to connect PCs forming a network, using Universal Software Radio Peripheral (USRP). The USRP created by Ettus Research [1], is a radio frontend that is commonly used with GNU Radio. Although the option of using a sound card provides a low cost solution, the USRP offers a wider frequency range as well as more dedicated hardware. The USRP has a total of 4 ADCs and 4 DACs allowing for up to 16 MHz of bandwidth each way, which is proficient for the underwater acoustic channel.

## 2.2 TinyOS and TOSSIM

TinyOS is a widely used sensor network operating system created at University of California, Berkeley and meant for sensor nodes requiring concurrency and flexibility while being limited to resource constraints [9]. TinyOS is implemented in the NesC language, which supports the concurrency model needed for sensor networks. TinyOS is widely used both in commercial applications as well as in academics for research purposes. There is a large user base who constantly contribute to the open source project, which allows UANT to always benefit from the newest protocols. For instance Washington University contributed a MAC Layer Architecture [7] to provide the MAC layer with many commonly needed functions for MAC protocols.

Generally TinyOS is compiled for a sensor network platform, with a network stack in accordance with the specific radio being used. However to gain the benefits of using TinyOS in UANT we have chosen to use TOSSIM [8], which simulates sensor network nodes on a PC. TOSSIM was created in order to support compiling TinyOS component graphs into a simulation for a PC. It utilizes a discrete event queue, reimplements some of the sensor nodes hardware, models the

radio and ADC of a mote, and most importantly for UANT, uses communication services for external programs to communicate with the running simulation over TCP sockets [8]. TOSSIM fully supports the TinyOS toolchain making it effortless to transition from simulation to real network, or vice versa. TOSSIM executes in a similar fashion to a real mote, with each simulation having a notion of a virtual clock running at 4MHz. TOSSIM uses interrupts similar to that of a mote, however when an event fires, the simulation calls an interrupt handler in a hardware abstraction component. In UANT, TOSSIM is not used as a simulation framework but rather the real time running nodes, this is explained in greater detail in the next section.

## 2.3 Current Underwater Solutions

### 2.3.1 MAC Layer

Due to the complications that the underwater acoustic channel provides, many of the current state of the art protocols that are used in terrestrial wireless communications, are not suitable underwater. One of the most notable differences between the two mediums is the propagation delay. This propagation delay that can often be neglected in terrestrial MAC protocols, is apparent underwater and must be taken into consideration. One MAC protocol that Syed et al. proposed for underwater acoustic communication is T-Lohi [18] which was designed to help ease the space-time uncertainty that is prevalent in the underwater acoustic channel. T-Lohi uses a contention based scheme where nodes send out a tone during a reservation period, and depending on the amount of contenders will either be followed by data transmission or another reservation period. Here the long propagation delay is compensated for as long as the reservation period is longer than the maximum propagation delay in the network.

Other suggested underwater acoustic MAC protocols include slotted FAMA [11] and UWAN-MAC [13]. UWAN-MAC simultaneously tries to overcome the long propagation delay as well as reduce energy to minimize cost. This is done by assuming that the propagation delay will not change between transmission cycles, then sleep-wake cycles are established for all nodes that are in communication range. Slotted FAMA [11] is an adaptation of FAMA in order to cope with the long propagation delay. In traditional FAMA, control packet lengths depend on the propagation delay which can lead to a large waste of energy for underwater acoustic networks. Slotted FAMA, similar to the of T-Lohi reservation periods, requires that each transmission slot be as long as the maximum propagation delay plus the length of a control packet. If a node wants to send data, first it will send an RTS at the beginning of a slot. If the receiving node allows the data transmission it will then send a CTS packet immediately following the RTS slot. Once the RTS/CTS exchange is complete, the transmitting node will begin to send the data two slots after the initial RTS was sent out. Finally an ACK is sent back to the transmitter to complete the exchange. Although this method will drastically reduce the amount of packet collisions, it comes at a cost of lower throughput due to the overhead associated with the protocol.

### 2.3.2 Physical Layer

Over the history of underwater acoustic communication nearly every form of modulation has been attempted. Phase, fre-

quency, amplitude, and spread spectrum techniques have all been used [6] [3]. While some of this variation can be attributed to the advancement of demodulation techniques, it is still not clear if there is one “best” PHY layer scheme, again this is due to the fact that there is no “typical” channel model.

The absence of a clear “winner” in both the physical and MAC layers, is the motivation for having a highly configurable system. UANT tries to provide this configurability which is needed in order to cope with the harsh underwater acoustic channel, while providing a full network to characterize application performance.

### 3. SYSTEM ARCHITECTURE

UANT has been created as a platform to allow testing of new protocols and modulation schemes on a fully functional underwater network. We used open source software (GNU Radio and TinyOS) that is widely supported and has been in use for many years. TinyOS applications are able to be incorporated into UANT simply by changing the configuration file to match the needs of the specific application. The running application can be easily monitored in Linux either by monitoring the TOSSIM simulation through currently supported techniques such as debug statements, or packets being forwarded over a TCP socket where the raw bytes of transmitted or received packets can be viewed, or both. Along with running TinyOS applications, it is possible to assign an IP addresses to the TOSSIM node in order to inject TCP/IP packets and use a wide variety of Linux applications to send data between nodes. UANT uses TOSSIM to run TinyOS applications and components on a PC in real-time fashion rather than being used as a simulator. This is similar to EmTos [5] where a wrapper around a TinyOS application is used to run on an EmStar node. It is important to note that each node in UANT is running in a different TOSSIM simulation. This allows the ability for every node to run a unique and different application, which is not possible in a single TOSSIM simulation.

Although the non-negligible latency introduced from samples traveling from the hardware frontend to software can make it difficult to develop a MAC layer for SDR, the long propagation delay of the underwater channel makes it possible to ignore this latency. In recent work from Nychis et al. [12] the use of “split-functionality” was implemented in order to take advantage of the minimal latency on the USRP, yet still maintaining the control of the data flow in the host CPU. However, with the slow propagation delay of the underwater acoustic medium, the added bus latency between the USRP and the MAC layer implemented on the host machine can be considered negligible. The speed of sound underwater is 1500 m/s, so for two nodes that are 1.5km apart the propagation delay is approximately one second. In [15], Schmid et al. measured SDR latency and characterized the round trip time to and from the USRP to be between 3ms and 26.5ms. The 26.5ms measurement included host processing of the 802.15.4 protocol, while 3ms is the theoretical latency to and from the host assuming no computation is done on the host machine. Although in terrestrial radio this latency makes some MAC protocols such as TDMA hard to implement, for underwater acoustics this is .3% of the propagation delay for nodes 1.5km apart.

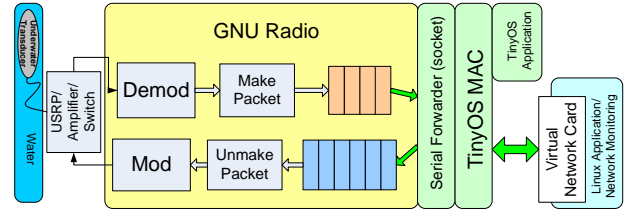


Figure 1: UANT System Architecture

### 3.1 Architecture

UANT provides reconfigurability at the physical layer, the MAC layer, and the application layer. With control over these layers, we aim to provide a way to implement and compare proposed underwater MAC protocols and PHY layer techniques in the presence of a network running applications from Linux or TinyOS. Although UANT must be run on a PC, we tried to keep in mind that the end goal for much of the current underwater acoustic communication research is geared toward underwater sensor networks (USN) and underwater autonomous vehicles (UAV).

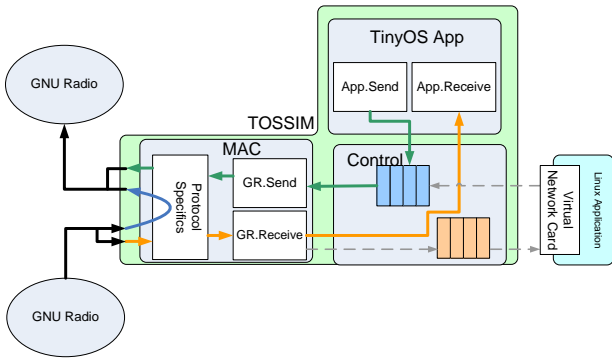
#### 3.1.1 Overview

The flow of data through UANT can be seen in Figure 1. To receive packets, data is first heard by the transducer and passed through to the USRP, which will sample and down convert the data. The data continues over USB to software where GNU Radio first passes the data through an FFT filter. The streaming samples now represent the complex baseband signal in the frequency domain needed for demodulation. Depending on the modulation scheme that is being used the data will be processed accordingly. The output of the demodulation blocks is a string of bits which will be fed to a message queue in GNU Radio.

GNU Radio is started with a Python script, which establishes the flow graph of C++ blocks for the data to travel through. In UANT, the same Python script that is used to start the flow graph is also used to communicate with TinyOS. Data in the message queue that has been demodulated and ready to enter TinyOS must first be packed into a TinyOS serial packet with a one byte modification. If the incoming packet has a broadcast address it is changed to be the address of the currently running node ID on the host machine, which is necessary due to the limitations of the SerialActiveMessages of TinyOS. The packet is then sent over a socket to TOSSIM where it first enters the MAC layer. Depending on what type of message has been received, it will be properly routed through the TOSSIM simulation, and possibly to Linux via a virtual Ethernet card if required.

#### 3.1.2 Physical Layer

GNU Radio has been widely adopted by the SDR community. There has been much advancement in terrestrial radios with the introduction of software defined radio. We have chosen to use GNU Radio for UANT in order to try and bring its success for fast prototyping and development to the underwater environment. GNU Radio offers many signal processing blocks that are required for basic and advanced modulation techniques. There have already been



**Figure 2: TOSSIM simulation on node**

many modulation schemes implemented in GNU Radio using these standard signal processing blocks. We have incorporated much of the prior work that has been done with GNU Radio including these modulation schemes as well as some of the example applications. Specifically we modified the tunnel example file that allows the USRP boards to be assigned IP addresses and use Linux applications. Modifications that were made include removing the simplified version of a single CSMA MAC, allowed for slower bit rates that could be desired in a harsh underwater channel, and changed the traffic source and sink from Linux to TinyOS. Keeping the modifications of the GNU Radio example to a minimum allows UANT to use any future expansion to the digital communication examples to be effortlessly incorporated into our system.

Although currently GNU Radio is too computationally expensive for embedded implementations, we preferred the ability for rapid prototyping and test of PHY and MAC even if it means that our host must be more powerful than some lower end platforms such as motes. If development of network protocols leads to increased performance on UANT it can ultimately be ported to the platform that is used in the USN or UAV, or directly transferred if TinyOS or Linux is used by the system. In addition, we later show that even with a more capable host UANT is still field-deployable and has been tested in different real world environments. Historically much of the development for GNU Radio has not considered energy savings which is essential for USN; however, there has been some recent work in this direction. In [12], there is a power saving method presented for fast pattern detection in order to only pass the received packet over USB if it is in fact intended for the listening node.

### 3.1.3 MAC Layer

One aspect of GNU Radio that makes it inherently hard for compatibility with higher layer protocols is the use of flow graphs. We have chosen to use TinyOS to implement the MAC layer and transition from GNU Radio flow graphs to a packet based system. In [10], Click was used to develop a MAC layer for SDR however all applications had to be run in Linux. The reason for using TinyOS in UANT is two-fold. First and foremost, much of the motivation to advance underwater acoustic communication comes from the attempt to further underwater sensor networks. USN have limited resource constraints and are the perfect candidate

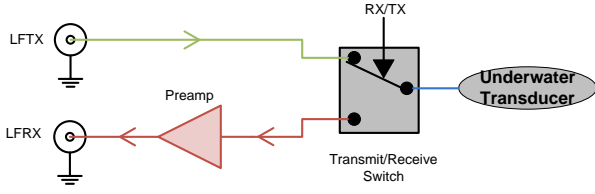
to run TinyOS since it is made specifically for this class of device. UANT is designed for research purposes specifically on different MAC and PHY layers, but using TinyOS will allow for an easier transition to an underwater sensor network node. The second reason we chose TinyOS comes from the fact that there is a large contributing base of users. This allows for UANT to benefit from development of protocols and algorithms for MAC and higher layers that are currently implemented as well as being developed for TinyOS.

As previously mentioned, GNU Radio provides a message queue which is able to make the transition from a streaming flow of data to a packet which highly facilitates the implementation of a MAC layer. In UANT the MAC layer sits closest to GNU Radio, in this way any packet that is sent to TOSSIM over the serial connection originating at the transducer will first pass through the MAC. A block diagram of the TOSSIM simulation can be seen in figure 2. If the packet has been determined to be a control packet, it will not proceed further than the MAC layer, and will be acted upon immediately, for instance to send an ACK, CTS, or any other packet needed for the current protocol in use. However, if the packet is for the running application, it will be sent straight through the MAC and control layers to the application. The received message is not queued here, it is left up to the application to queue received messages if needed. Messages that are to go beyond the MAC layer are only queued if they are intended for Linux via the virtual Ethernet card.

### 3.1.4 Application Layer

UANT not only provides flexibility at the MAC and PHY layer, but also gives two options on where to run applications. TinyOS applications that can currently run in TOSSIM, can be easily integrated into UANT. The configuration file must be wired properly in order to correctly connect the applications components, while the application code itself will change very minimally if at all. Using TinyOS in UANT for applications allows to develop in an environment similar to what could potentially run on an USN. These applications along with the MAC layer could be directly implemented on a sensor node running TinyOS in the future. For more complex applications we have also provided the ability for Linux to be used. In UANT if this option is enabled, a virtual Ethernet card will be established. All IP addresses of the network must have the same network ID and the possible number of unique addresses for the network is 254. Applications such as ping, file transfer, and even media streaming applications could be possible with the use of proper physical layer modulation schemes that allow for the bandwidth needed. Not only does this allow for Linux applications to be used, but also for network monitoring tools that have been created such as Wireshark [2] to be used to help characterize and monitor the links, since Linux sees UANT as a NIC.

It is possible to run both Linux and TinyOS applications simultaneously. This is accomplished using a control block that sits in TinyOS between the MAC and the application. A FIFO queue is kept (of configurable size) which allows packets from TinyOS and Linux to be multiplexed for more flexibility. When the MAC is ready to send a packet from a higher layer, it will simply signal for the first message in the queue. Any pending packets in the queue will be sent out



**Figure 3:** A simplified schematic of preamplifier/switch board showing bidirectional nature of using single underwater transducer and a switch controlled by USRP. Transmit path (green), receive path (red), and shared transmit and receive path (blue) are shown.

to GNU Radio and through the water.

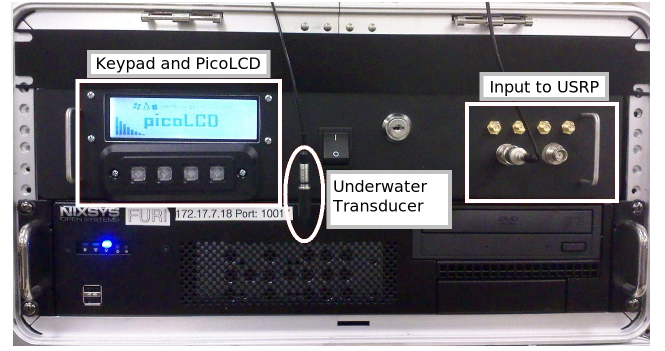
### 3.2 UANT and Linux

If Linux applications are chosen to be used then UANT benefits from Linux's TCP/IP stack. This is accomplished by using a TUN driver. A Python script is used to help establish the virtual Ethernet card and manage packets between Linux and TinyOS. Once the Python script has been started, an IP address can be assigned to the TinyOS node. The IP address that is manually assigned must have the same network address as all other nodes and the last byte must correspond to the same node ID used for the TOSSIM simulation. One current limitation in UANT regarding packets originating from Linux is the maximum transmission unit (MTU) must be set below 247 bytes. The reason for this is the packet size from Linux plus the TinyOS header size must be less than 256 bytes in order to be used with TinyOS packet protocols. The good news is UANT assigns the appropriate IP address with the MTU size that is set by the user.

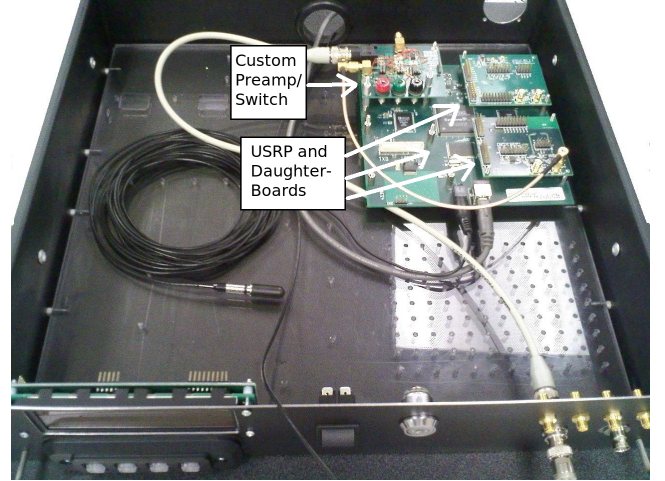
Once the TinyOS node has been assigned an IP address Linux views UANT as a network card. The advantage to this approach is that UANT can now provide a full networking solution for underwater acoustics using the TCP/IP stack of Linux. As we later show this allows for application performance to be studied as a function of configuring the lower layers including MAC and PHY. If TinyOS applications are preferred than the many networking protocols that are available and used for sensor networks can be easily incorporated into UANT.

## 4. IMPLEMENTATION

We have implemented UANT on Linux boxes using the Intel core 2 quad Q8200 processor, with 4GB of memory. We used USRPs using the LFTX and LFRX daughterboards which have a range of DC-50MHz, allowing for use in any underwater acoustic range. Typically with the low frequency daughterboards a separate transducer must be used for receiving and transmitting, since the daughterboards are independent. We have built a custom preamplifier board that also incorporates a switch in order to allow for one transducer per node, as well as amplify the received signal entering the USRP. We used RESON TC 4013 transducer for both transmitting and receiving, with a possible range of .1 Hz to 175 kHz.



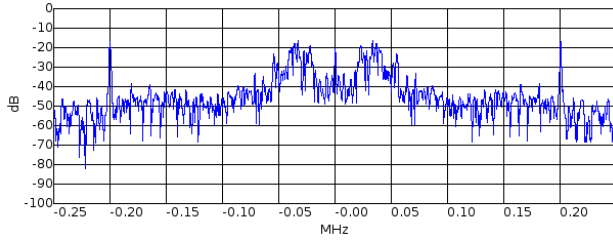
(a) Front view of UANT system



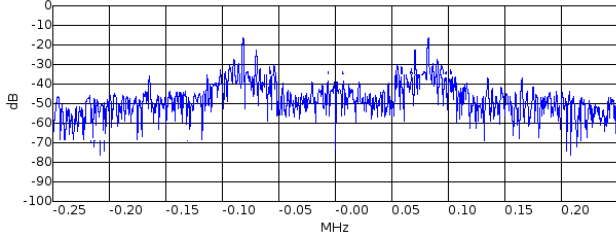
(b) View inside drawer

**Figure 4:** These two figures show a deployable UANT system (a) front view and (b) inside view of USRP, low frequency daughterboards and custom preamplifier and switch.

The preamplifier that we built cut cost by using off the shelf ICs. Furthermore only one transducer is needed rather than the usual two transducers required for the low frequency USRP daughterboards. The USRP allows control over several GPIOs which we wired to a switch on the preamplifier board. On one end of the SPDT switch is the transducer being used both for transmitting and receiving. The receive side of the switch is fed into the preamplifier and finally out to the LFRX daughterboard. While the other signal path begins at the LFTX daughter board, goes through the switch and out the board to the transducer and through the water. A simplified schematic of this can be seen in figure 3. We have configured the GPIO to remain high unless the USRP is transmitting in which case it will be toggled low. After the transmission is completed the GPIO will again be high and the system will be in receive mode waiting for an ACK or any other packet to receive. The preamplifier can be powered from the USRP which ensures the need of only one power source for all components of UANT, making field deployments much easier. Powering the preamplifier from the USRP allows for a maximum voltage swing of 6.6V. Using a high gain amplifier we were able to correctly demodulate a signal less than 10 mV without any problem.



(a) FFT of tank setup.



(b) FFT of larger pool.

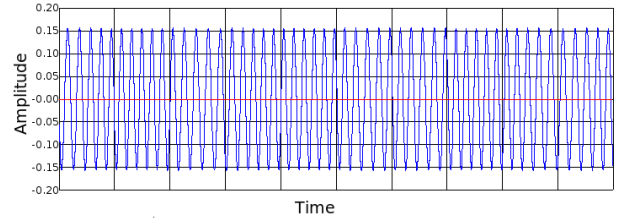
**Figure 5: Frequency spectrum of tank (a) and pool (b). The two large noise sources (around 25kHz in the tank, and 75kHz in the pool) are the water pumps.**

A deployable UANT system can be seen in figure 4. We have used rackmount servers (described above) enclosed in a rugged case for UANT to be taken into various terrain. A rackmount drawer was modified for UANT to be able to quickly access all connections to and from the USRP and the outside world. The USRP can be powered with any 5-6 V DC power supply allowing hard drive power connectors to power the USRP along with our amplifier boards used for acoustic communication. For field deployments a universal power supply can be used to power these UANT boxes.

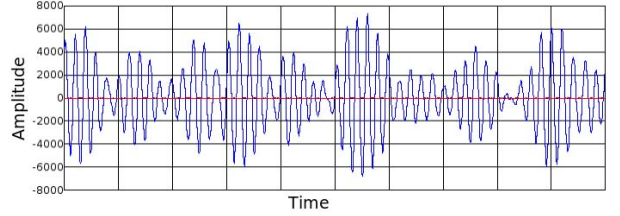
For our MAC layer we currently implemented a basic Aloha protocol in TinyOS. The configurable parameters for this MAC are the minimum and maximum times for a back-off interval. If no ACK was received after data was sent, a retransmission is sent after a random time between the provided interval.

For the physical layer we have taken advantage of many of the configurable options that are provided for digital communication in GNU Radio. Transmission bit rate can be configured from 244 bits per second to 500 kilobits per second. We also varied the center frequency, and the limits are from .1 Hz up to 30 MHz which makes the real limiting factor the transducers (and the channel) being used which in our case was a maximum of 175 kHz.

Currently, due to transmission output power limitations (less than 500 mW) we have been limited to the distance of our communications. The USRP has a maximum output voltage of two volts peak-to-peak. The underwater transducers being used have high input impedance at ideal acoustic frequencies requiring further power amplification for long range. For most of our testing we used high frequencies (100 kHz - 175 kHz) because the impedance of the transducer is much lower allowing for higher transmit power. The



(a) Original GMSK signal without multipath.



(b) Effect of multipath on a GMSK signal.

**Figure 6: These two figures show the effect of multipath on a GMSK Signal. (a) shows the original signal, without multipath. Figure (b) is the result if the same signal gets transmitted through a tank.**

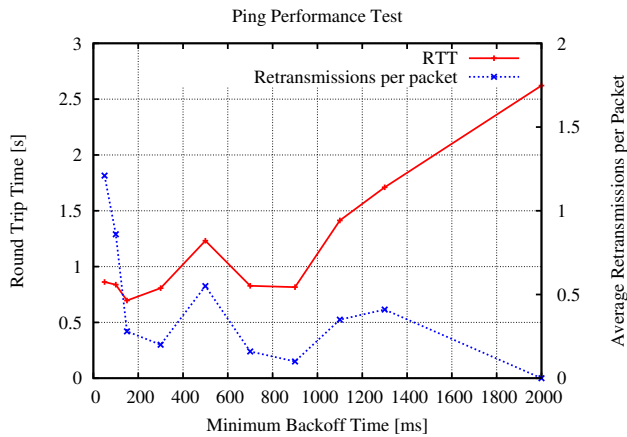
gain of this added transmit power was more beneficial than the higher attenuation with distance that occurs at these frequencies. Even with the power limitation we have still successfully deployed UANT both in our lab tank as well as in a pool.

## 5. EVALUATION

The need to have a deployment time configurable system is realized with figure 5. Depending on the location the optimal transmit center frequency can differ. The noise source of the lab tank was primarily the filtration system which was centered around 25kHz. When we deployed UANT in the pool we found that the noise source was centered closer to 75kHz.

Although it might be expected that a controlled environment would produce better results than a real world scenario this is not always the case. We found that our tank inside the lab was unfavorable for acoustic communication. The reason for the poor results can be seen in figure 6, where the multipath of the signal is quite apparent. The modulation for this test was not amplitude modulation but rather GMSK. The reason there is such a variation in the amplitude of the signal is multipath and inter-symbol interference, which is a result of the slow speed of propagation. The best remedy for the multipath in our lab tank is to lower the transmission speed, this allows more samples per symbol to be sent out to the USRP and received at the transducer.

One counterintuitive result we found in our lab tank testing was the effect of our water filtration system. Generally introducing a source of noise such as a filter that produces bubbles, acoustic noise, and surface waves, will degrade performance. Surprisingly this is not the case and the explanation goes back to multipath. With the water standing still a transducer that is attempting to receive can be caught in a

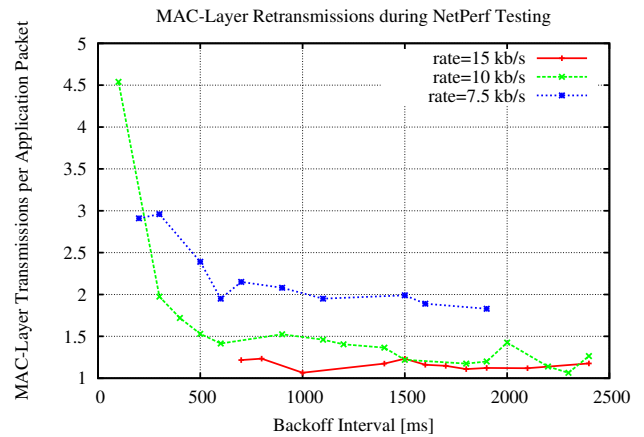


**Figure 7: Average number of packet retransmissions needed and round trip time with varying minimum backoff time. Ping performance averaged over 50 pings for each data point with center frequency at 100 kHz and transmit rate at 5 kb/s. All backoff intervals (max - min) set to 400ms**

blind spot of destructive multipath interference such that almost no signal is seen. Although typically these blind spots vary with time and space due to moving reflective surfaces such as waves, in a controlled environment this is potentially not the case. Turning on the filter, although introducing noise, improved system performance by receiving up to twice the amount of correct packets because of the now time varying blind spots.

UANT allows the ability to see application level performance as a factor of the tunable parameters at the PHY and MAC layer. Figure 7 shows how changing the minimum time needed to wait for an ACK before retransmission affect the Linux ping application. The backoff interval is set to 400 ms for this experiment and all other parameters are constant. If the minimum time to wait is too small there will be many packet collisions mostly with the retransmission of data and the ACK packet itself, this leads to an increased number of retransmissions. On the other hand, if the minimum time before retransmission is too large than if a packet is not received the transmitter is waiting needlessly long to resend the data leading to low throughput and a high round trip time (RTT) even though very few retransmissions are needed. Looking at this plot it is clear that trade-offs can be achieved to either optimize for minimum transmissions, quickest RTT, or a minion of both.

Another application that was tested in UANT was Netperf. Netperf establishes a connection between a client and server and then sends TCP packets across the link. Figure 8 shows the results from the Netperf testing, where each data point was collected over approximately a 300 second interval. Here we varied the transmission rate at both the physical layer and MAC layer to see how that would affect performance. Regardless of the transmission rate if the Aloha minimum backoff interval was too low than it would cause many retransmissions. As the minimum backoff interval increased we found that the amount of retransmissions



**Figure 8: Netperf performance while varying minimum Aloha backoff time at different transmission rates. Each data point collected over 300 seconds of communication**

decreased since the nodes were not trying to retransmit too soon. As the transmission rate varied the amount of retransmissions needed also varied. The fastest rate of 15 kb/s lead to the least amount of retransmissions while the slowest rate that was tested (7.5 kb/s) resulted in the highest amount of retransmissions. This result brings up another advantage of having an easily reconfigurable system which allows to specify parameters at runtime.

We have also implemented an example TinyOS application that fires a periodic timer and transmits its count over the network to the other nodes. We were able to leave the application code unmodified while only changing the wiring configuration. The throughput of this application was much higher than the Linux applications that we tested for performance. One reason for this being that the overhead of a TCP/IP packet is much greater than the eight bit TinyOS message header that was used. Also there is added latency for packets from Linux to get to TinyOS and then sent out, when compared to that of the application currently running in TOSSIM.

## 6. FUTURE WORK

Thus far UANT has benefited from widely used open source tools namely; GNU Radio and TinyOS. Although we have applied their protocols and algorithms along with implementing a few of our own, many are geared toward over the air or wired communication. In the future we plan to implement some of the currently proposed underwater protocol solutions mentioned in section 2. Allowing the protocols to be compared using UANT with either a TinyOS or Linux application to benchmark performance will help distinguish strengths of each solution as compared with one another. We plan to further study the effect of changing PHY and MAC layer parameters.

Along with building the repertoire of protocols and configurable options we plan to take advantage of the software defined approach that is used in UANT. For instance, currently all the parameters are set at deployment time and if

loud noise sources arise at the chosen center frequency performance will degrade. In the future we plan to incorporate adaptively setting parameters such as frequency band and transmit power based on channel conditions, distance, noise and interference. In this way UANT will be able to cope with many undesirable effects and still have good performance. We also plan to continue to study the trade-offs that are apparent with different MAC and PHY layer schemes.

## 7. CONCLUSION

The challenges of the underwater acoustic channel are great, requiring the need for flexibility during system development. We have presented UANT as a complete end-to-end networking platform for underwater acoustic communication. UANT is designed for field-deployment, geared toward fast prototyping and testing of PHY and MAC layer schemes. It is implemented using open source software, facilitating further extension. We have demonstrated a Linux application running on UANT and shown how configuring the system can lead to a benefit in overall performance.

## 8. ACKNOWLEDGMENTS

This material is supported in part by the U.S. Office of Naval Research under MURI Award CR-19097-430345, and the UCLA Center for Embedded Networked Sensing. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the listed funding agencies.

## 9. REFERENCES

- [1] USRP brochure. <http://www.ettus.com/>, April 2009.
- [2] Wireshark network protocol analyzer. <http://www.wireshark.org/>, April 2009.
- [3] I. F. Akyildiz, D. Pompili, and T. Melodia. Underwater acoustic sensor networks: Research challenges. *Ad Hoc Networks Elsevier*, 3:257–279, 2005.
- [4] E. Blossom. Exploring GNU radio. <http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html>, April 2009.
- [5] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer. A system for simulation, emulation, and deployment of heterogeneous sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 201–213. ACM, 2004.
- [6] E. Jones. The application of software radio techniques to underwater acoustic communications. pages 1–6, June 2007.
- [7] K. Klues, G. Hackmann, O. Chipara, and C. Lu. A component-based architecture for power-efficient media access control in wireless sensor networks. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 59–72, New York, NY, USA, 2007. ACM.
- [8] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, NY, USA, 2003. ACM.
- [9] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, E. B. M. Welsh, and D. Culler. *Ambient Intelligence*, pages 115–148. Springer Berlin Heidelberg, 2005.
- [10] K. Mandke, S.-H. Choi, G. Kim, R. Grant, R. Daniels, W. Kim, R. Heath, and S. Nettles. Early results on hydra: A flexible mac/phy multihop testbed. pages 1896–1900, April 2007.
- [11] M. Molins and M. Stojanovic. Slotted fama: a mac protocol for underwater acoustic networks. pages 1–7, May 2006.
- [12] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, and P. Steenkiste. Enabling MAC protocol implementations on software-defined radios. 2009.
- [13] M. K. Park and V. Rodoplu. Uwan-mac: An energy-efficient mac protocol for underwater acoustic wireless sensor networks. *Oceanic Engineering, IEEE Journal of*, 32(3):710–720, July 2007.
- [14] J. Preisig. Acoustic propagation considerations for underwater acoustic communications network development. *SIGMOBILE Mob. Comput. Commun. Rev.*, 11(4):2–10, 2007.
- [15] T. Schmid, O. Sekkat, and M. B. Srivastava. An experimental study of network performance impact of increased latency in software defined radios. In *WinTECH '07: Proceedings of the the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, pages 59–66, New York, NY, USA, 2007. ACM.
- [16] E. Sozer, M. Stojanovic, and J. Proakis. Underwater acoustic networks. *Oceanic Engineering, IEEE Journal of*, 25(1):72–83, Jan 2000.
- [17] E. M. Sözer and M. Stojanovic. Reconfigurable acoustic modem for underwater sensor networks. In *WUWNet '06: Proceedings of the 1st ACM international workshop on Underwater networks*, pages 101–104, New York, NY, USA, 2006. ACM.
- [18] A. Syed, W. Ye, and J. Heidemann. T-lohi: A new class of mac protocols for underwater acoustic sensor networks. pages 231–235, April 2008.